KISSZÁMITÓGÉPEK A KÖZÉPISKOLÁBAN

/Egy aritmetika felépitése és néhány alkalmazása/

Dr. Csuri József

I. *Bevezetés*

Közismert és általánosan elfogadott, hogy a számitógépek felhasználása az élet szinte minden területén rendkivül nagy jelentőségü. Ebből a - sajnos elég késői - felismerésből eredően ma már kormányprogram szorgalmazza a számitástechnikai kultura széles körü elterjesztését, a számitógépek minél hatékonyabb felhasználása érdekében. Ebben a munkában - sok más intézményekkel, szervezetekkel együtt - nagyon fontos szerepet tölthetnek be a középiskolák is.

A középiskolában a számitógépek többféle összefüggésben szerepelhetnek: segédeszközt jelenthetnek az iskola ügyvitelében, valamint az egyes tárgyak oktatásában, s ugyanakkor az oktatás tárgyát is képezhetik. A legközelebbi jövőben a számitógép oktatást segitő, támogató szerepének - számitógéppel segitett oktatás - lesz legnagyobb jelentősége, ez azonban természetszerüen összefügg a gépek müködtetésének, programozásának oktatásával is.

A számitástechnikai kultura elterjesztésének *anyagi* és személyi *feltételei* vannak. E tekintetben történt már némi előrehaladás. A korábbi néhány évben a középiskolák - ha korlátozott számban is - beszerezhettek programozható zsebszámológépeket. Ezek a gépek, programozhatóságuknál fogva, ha nem is elég széles körben, hozzájárultak ahhoz, hogy az érdeklődőbb tanulók némi betekintést kaphattak a számitástechnika elemeibe. Ezek a gépek - természetükből kifolyó-

lag - elsősorban csak a természettudományos tárgyak igényeit elégithették ki, sajnos azonban sok helyen még ezeket a lehetőségeket sem használták ki eléggé. További jelentős eredmény, hogy ebben az évben minden középiskola hozzájutott legalább egy kisszámitógéphez. Ugyancsak nagy jelentőségü, hogy több középiskola terminálon keresztül nagyszámitógéphez csatlakozhat.

A kis- és nagyszámitógépek minőségileg uj lehetőségeket nyujtanak a középiskola valamennyi tárgyának oktatásában.

Történtek kezdeményezések a *személyi feltételek* megteremtése érdekében is. Több egyetem tanárszakos hallgatói kiegészitő szakként tanulhatják a számitástechnikát, s megnyugtató, hogy - főleg a jobb képességü hallgatók körében - komoly az érdeklődés az ilyen képzés iránt. Ez a képzési forma rövid távon mindenképpen szükséges, de talán hosszu távon is célravezetőnek tünik. A jelent és a közeljövőt illetően a legtöbb középiskolában szükségmegoldáshoz kell folyamodni. A számitástechnikai ismeretek terjesztését, tanitását a matematikával való szerves és szoros kapcsolata folytán - ha nem is kizárólag, de - elsősorban a matematika tanárainak kell vállalniuk, részben tárgyuk keretében, részben azon kivül. A távolabbi jövőt illetően szükségesnek látszik, hogy erre a célra a matematika óraszámát valamelyest megnöveljük, anélkül, hogy ez a tanulók összóraszámát növelné. Ez a megoldás kölcsönös motivációt jelent a matematika és a számitástechnika számára, s ezen tul a többi tárggyal is elmélyült koncentráció lehetőségét biztositja. Természetesen a számitástechnikai müveltség terjesztését nem lehet, s nem szabad néhány tárgy keretére leszükiteni. Ebből a szempontból - az oktatás hatékonyságának növelésén tul is - rendkivül nagy jelentőségü a gépeknek a különböző tárgyak oktatásában betöltött segédeszköz szerepe. Hogy a gépek ezt a szerepet hasznosan betölthessék, a középiskolai tanároknak mielőbb el kell sajátitani a kezelésüket - olyan

szinten, hogy kész programokat tudjanak futtatni. A hatékonyság növelése céljából egy későbbi fokozatban - néhány év elteltével - a tanároknak, a gépek kezelésén tul, legalább olyan mértékü programozási ismereteket is szerezniük kell, hogy alkalmas szubrutinokból össze tudjanak szerkeszteni egyéni elképzeléseiknek, az egyes osztályok adottságainak megfelelő programokat. A tanárok képzésére fel lehet használni a szokásos továbbképzési alkalmakat és megfelelő irodalom közreadásával biztositani kell önképzésük lehetőségét. Elengedhetetlenül fontos, hogy a gépek használati utmutatói, a programozásukhoz szükséges alapvető ismeretek, példatárak tanárok és diákok számára egyaránt hozzáférhetők legyenek. Ezek a kiadványok a szakköri munkát is jól segithetik.

A számitástechnikai ismeretek széles körü középiskolai felhasználása és tanitása terén még sajnos kevés a hazai tapasztalat. Valamivel bővebb a külföldi tapasztalat, de az sem eléggé hozzáférhető, és a mi viszonyainkra nehezen is alkalmazhatók. Megfelelő tapasztalatok gyüjtése céljából az iskolák munkaközösségei és az egyes tárgyakat jól ismerő számitástechnikai szakemberek is vizsgálják meg a különböző középiskolai tárgyak tantervi anyagát abból a szempontból, hogy melyek azok a témakörök, anyagrészek, amelyek tanitásában ésszerüen, hatékonyan fel lehet használni a számitógépeket. A vizsgálatok összegezése után mielőbb meg kell iratni - lehetőleg képzett számitástechnikai szakemberekkel - a szükségesnek, hasznosnak vélt programokat, s azokat mielőbb a középiskolák rendelkezésére kell bocsátani. Ugyancsak sürgősen el kell dönteni - a számitástechnikai szakemberek bevonásával - hogy melyek azok a legalapvetőbb, számitástechnikai ismeretek, amelyeket minden középiskolai tanulónak el kell sajátitania ahhoz, hogy a mind gyorsabban fejlődő számitógépes környezetbe be tudjon illeszkedni. Körül kell határolni a célravezetőnek látszó kereteket és módszereket is.

Megjegyezzük, hogy bizonyos mértékü programozási ismeretek tanitására is feltétlenül szükség van, ugyanis éppen az jelenti a legnagyobb hajtóerőt, hogy a tanulók konstruktiv módon be tudnak avatkozni a gépek müködésébe, hogy saját elképzeléseiket tudják a gép segitségével megvalósitani. Tapasztalatok mutatják, hogy a tanulók többsége erőteljesen igényli a programozási ismereteket.

A számitástechnikai kultura megalapozása néhány aktuális kérdésének felvetése után a továbbiakban egy konkrét anyagon keresztül próbáljuk bemutatni, hogy alapvető müveleteket végző rutinokból hogyan lehet programokat összeállitani, s egyben a kisszámitógépek számábrázolási módjából adódó bizonyos hátrányokat igyekszünk kiküszöbölni.

II. *Az aritmetika előkészitése*

Mint ismeretes az ABC-80 tipusu kisszámitógép az egész tipusu változókat 16 biten, kettes komplemens kódban ábrázolja, igy bennük csak a $[-2^{15}, 2^{15}-1]$ intervallum egész számai férnek el. A valós tipusu számok mantisszái maximálisan 6 értékes jegyüek lehetnek /a 10-es alapu számrendszerben/. Rendelkezik a gép ASCII aritmetikával is, amely a maximálisan 29 karakteren ábrázolható, valós tipusu számok körében tudja elvégezni az alapmüveleteket, feltéve, hogy a végeredmény is elfér 29 karakteren. /A hatványozást nem végzi el!/ A HT-1080Z iskolai kisszámitógépek is hasonlóan ábrázolják az egész tipusu változókat, a valós tipusu számok ábrázolása egyszeres pontosságu változóban max. 6, kétszeres pontosságu változóban 16 értékes jegyig terjed, ASCII aritmetikája nincs. Az összes eddig felsorolt esetben kerekités nélküli, teljes pontosságu végeredmény csak akkor kapható, ha mind az operanduszok, mind a pontos eredmény ábrázolható a megfelelő tartományban.

A számelméletben, a kombinatorikában de nagyon sok más témakörben is gyakran előfordul, hogy viszonylag kis kiin-

dulási adatok esetén is rendkivül nagy rész- ill. végeredmények adódnak, amelyek pontos, /kerekités nélküli/ számitását a gépek számábrázolási lehetőségei korlátozzák. Sok esetben pedig elengedhetetlen a teljes pontosság, hiszen különben teljesen értelmetlenné és igy feleslegessé válhatnak a kapott eredmények. Nincs pl. értelme annak, hogy valamely egész szám primtényezős felbontását annak valamilyen kerekitett értékén végezzük el. Sokszor hasznos lehet egy-egy számnak vagy egy sorozat egyes tagjainak teljesen pontos, tényleges előállitása, vagy egy számnak tetszés szerinti hibahatáron belüli - esetleg pontos jegyekkel történő - közelitése.

E meggondolások alapján az alábbiakban az ABC-80 tipusu gép BASIC-nyelvén olyan szubrutinokat állitunk össze, amelyek segitségével a nemnegativ egész számok körében az összeadás, kivonás, szorzás, maradékos osztás és a hatványozás müveletét teljes pontossággal /kerekitési hiba nélkül/ tudjuk elvégezni ugy, hogy az operanduszok jegyeinek a számára - a futási időt és a gép memoriakapacitását leszámitva - semmiféle korlátozás nincs. /A kivonás eredménye negativ is lehet!/

Megjegyezzük, hogy a tizedes tört kitevőre való hatványozást leszámitva nem jelent lényeges megszoritást az a tény, hogy a szóbanforgó rutinok csak a nemnegativ egész számok körében tudják elvégezni a megfelelő müveleteket. A felhasználó az operanduszok előjelének és karakterisztikájának ismeretében könnyen kaphatja az eredmény előjelét és karakterisztikáját, s igy a tényleges müveletet már csak a nemnegativ egész számok körében kell végezni. Ebből következik, hogy a rutinok tetszés szerinti kis számokon is el tudják végezni a szóbanforgó müveleteket, s igy alkalmasak adott számok tetszés szerinti pontosságu közelitésére is.

A rutinok által megadott aritmetika alkalmazásaként elkészitünk néhány további szubrutint is, amelyek elsősorban a számelméletben ill. a kombinatorikában gyakran előforduló mennyiségek /legnagyobb közös osztó, ismétlés nélküli variá-

ciók száma, binomiális együtthatók/ számitására alkalmasak. További alkalmazásként még egy programot is készitettünk. Ez az ABC-80 tip. gép konstrukciójához alkalmazkodva, max. 119 jegyü egész számok törzstényezős felbontását végzi el. Ezzel kapcsolatban megjegyezzük, hogy a jegyek számára tett korlátozás könnyen megszüntethető, s hogy a program némi változtatással szubrutinná alakitható, s igy annak felhasználásával több, érdekes számelméleti függvény /primtényezők száma, osztók száma, osztók összege, stb./ értékei számithatók. Az igy kapható rutin módot ad sok érdekes számelméleti probléma vizsgálatára akármilyen nagy számok esetén is. /Pl. tökéletes számok, barátságos számok, erősen összetett számok, Mersenne-féle primszámok, Fermat-féle primszámok, stb./

A rutinokat, programokat az ABC-80 tip. gép BASIC-nyelvén irtuk, de azok egészen csekély módositással a HT-1080Z gépek számára is átirhatók.

A rutinok REM utasitásokban tartalmazzák a felhasználásukkal kapcsolatos legfontosabb tudnivalókat. Ezek a REM utasitások törölhetők is, ekkor azonban híváskor a megmaradó legkisebb sorszámu utasitásra kell hivatkozni.

A rutinokat ugy készitettük, hogy hivásuk előtt a bemenő adataik, amelyek nemnegativ egész számok, tizes számrendszerbeli jegyeit egy-egy aritmetikai tömb egyes elemeibe kell helyeznünk. Pontosan:

Az $A = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_1 \cdot 10 + a_0$

$0 < a_i \leq 9$, $i = 1;2;\ldots;n$ és $a_n < 0$

alakban felirt pozitiv egész számot ugy ábrázoljuk az A% *-gal jelölt aritmetikai tömbben, hogy a szám 10^i helyiértékü jegyét az A% * tömb i indexü elemébe helyezzük, tehát

$$aA\%\ i = a_i$$

n-et, vagyis a szám legnagyobb helyiértékü, 0-tól különböző jegyének tömbindexét a tömb azonositójának megfelelően /most tehát/ A%-kal jelöljük, s ezt is meg kell adni a rutin számára. /A% = n/

Az A = 0 szám ábrázolása az A% * tömbben az A% * = 0 és A% = 0 egyenlőségekkel történik.

Jól látható és nagyon fontos, hogy az alkalmazott számábrázolás nem engedi meg a vezető 0-ok használatát. A szám jegyeivel együtt mindig meg kell adni a legmagasabb helyiértékü pozitiv jegyének a tömbindexét is, ha ilyen van. A 0 szám esetén ez a tömbindex definiciószerüen 0. Ez az ábrázolási mód talán körülményesnek tünhet, azonban sok előnye van.

Az A = 25047 számot pl. az A%(*) legalább 5 elemü tömbben az A%(0) = 7; A%(1) = 4; A%(2) = 0; A%(3) = 5; A%(4) = 2 és A% = 4 egyenlőségekkel adjuk meg. Ugyanennek a számnak az ábrázolását szemléletesen is mutatja az alábbi ábra:

	A%(4),	A%(3),	A%(2),	A%(1),	A%(0)
A%(*)	2	5	0	4	7

A% = 4

A rutinok a kirenő adatokat is ilyen alakban adják meg, hogy azok ujabb átalakitás nélkül, a megfelelő változókba való áttöltés után bemenő adatai lehessenek valamelyik rutinnak /esetleg saját magának/.

A rutinok a bemenő adatokat nem változtatják meg, sőt arra is törekedtünk, hogy lehetőleg az őket tartalmazó tömbök se változzanak meg. Ahol ettől eltértünk, ott a leirásban erre utalunk.

A rutinok ill. programok müködését, a müködés alapjául szolgáló algoritmusokat leirtuk, s ahol szükségesnek véltük blokkdiagramon is szemléltettük. A megirásuk során szükségessé váló segédváltozók, tömbök azonositóit ugy választottuk meg, hogy olyan esetben sem következzen be hiba, ha az egyik rutin hivja a másikat. Arra is ügyeltünk, hogy olyan azonositókat használjunk a szubrutinokban, amelyeket - kényelmetlenségüknél fogva - programok irása során nem szokás használni /1/2%, 1/2%(*), stb./.

Az ABC-80 tipusu számitógép BASIC nyelvében lévő automatikus tömbdeklaráció szerint az aritmetikai tömböket csak

akkor kell deklarálni, ha méretük 10-nél nagyobb. A rutinok leírásában mindenütt utaltunk arra, hogy az egyes tömböket legalább mekkora méretüre kell deklarálni a bemenő adatok méretétől függően, figyelembe véve a számolás teljes folyamatát. Több helyen érthetően csak azt adtuk meg, hogy egy-egy tömbben mely számnak kell elférnie.

A használt rutinokban szereplő tömbök deklarálásánál figyelembe kell venni, hogy az ABC-80 BASIC-nyelvében a tömb ujradeklarálása esetén a tömb mérete nem növekedhet. A rutin többszöri hivása esetén ennek megfelelően kell a tömbök méretét megválasztani.

A rutinok ill. programok használatához tudni kell, - s ez hibájukként is felróható - hogy nem vizsgálják meg a bemenő adatokat, s esetleg hibás adatokon is elvégzik az előirt müveleteket, s esetleg hibás, vagy értelmetlen eredményt szolgáltatnak és nem adnak erről hibajelzést.

A rutinokat és programokat szalagon is tároltuk.

III. *Szubrutinok és programok*
/leirások, listák/

Nemnegativ egész számok összeadása
/ADDRUT/

Az itt leirt rutin az A%(*) ill. B%(*) tömbökben elhelyezett, akárhány jegyü, nemnegativ egész számok összegét számitja ki, s az összeget a C%(*) tömbbe helyezi.

A rutin az összeadás szokásos algoritmusát követi. Az egyes helyiértékeken adódó átvitel ideiglenes tárolására az +% változót használja. Ha a két szám jegyeinek száma nem egyenlő - tegyük fel, hogy pl. A% > B% - akkor a rutin a B%(*) tömb B%-nál nagyobb indexü elemeit az összeadás során figyelmen kivül hagyja.

Az A%(*), B%(*) és C%(*) tömböket - feltéve, hogy az automatikus deklaráció nem teszi feleslegessé - deklarálni

kell. C%(*) deklarálásánál ügyelni kell arra, hogy az összeadandó.

A rutin a 10000 sorszámu utasitással kezdődik, s a 10140 sorszámu utasitásig tart.

A rutinban az 1/2%, ←%, →% segédváltozók szerepelnek.

```
10000 REM RUTIN KET NEMNEGATIV EGESZ SZAM OSSZEADASARA.
10010 REM A TAGOKAT AZ A%(%) ILL. B%(%) TOMBOKBE KELL HELYEZ
      NI , A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELEMEBE.
10020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK A%
      ILL. B% TOMBINDEXET IS.
10030 REM AZ EREDMENY A C%(%) TOMBBE KERUL. A GEP MEGADJA C%
      -OT IS, AMINEK JELENTESE AZ ELOZOKBOL ERTHETO.
10040 REM A RUTIN AZ %% , ←% ES →% SEGEDVALTOZOKAT HASZNALJA
10050 IF A%<B% THEN C%=B% : →%=0% ELSE C%=A% : →%=1%
10060 ←%=0%
10070 FOR %%=0% TO C%
10080 IF →%=1% AND %%>B% THEN ←%=A%(%%)+←% : GOTO 10110
10090 IF →%=0% AND %%>A% THEN ←%=B%(%%)+←% : GOTO 10110
10100 ←%=A%(%%)+B%(%%)+←%
10110 C%(%%)=←%-←%/10%*10% : ←%=←%/10%
10120 NEXT %%
10130 IF ←%>0% THEN C%=C%+1% : C%(C%)=←%
10140 RETURN
```

Két nemnegativ egész szám kivonása

/SUBRUT/

Az alább leirt rutin az A%(*) ill. B%(*) tömbökben elhelyezett, akárhány jegyü, nemnegativ egész számok különbségét számitja ki, majd a különbség abszolut értékét a C%(*) tömbbe helyezi. Negativ különbség esetén a C$ változóba a "-" jelet tölti be, egyébként C$ üres lesz.

A rutin először megvizsgálja, hogy a két szám közül valamelyik nagyobb-e a másiknál, s ha igen, melyik az. Ha egyenlőek, akkor a 0 különbséget betölti a C%(*)tömbbe, és visszatér a főprogramba. Ha a kivonandó nagyobb a kisebbitendőnél, akkor elvégzi a C$ = "-" értékadást, különben C$ = "-". Ezt követően a nagyobb szám egyes jegyeiből ciklusban kivonja a kisebb szám megfelelő jegyeit, s azokat a C%(*) tömb megfelelő elemeibe helyezi. Ha a kisebb szám ke-

vesebb jegyü, akkor megfelelő számu vezető 0-t képzelünk eléje, ezt a helyettesitést a program valójában nem végzi el, hanem feltételes utasitással éri el, hogy a kisebbitendőnek a képzelt 0-kal azonos helyiértékü jegyei változatlanul kerüljenek a különbségbe. A C%(*) tömb elemeiben ekkor "negativ jegyek" is szerepelhetnek. Egy ujabb ciklus alakitja ki a különbség tényleges jegyeit megfelelő átvitelekkel. Egy további ciklus állapitja meg a különbség legmagasabb helyiértékü jegyének C% tömbindexét.

Az A%(*), B%(*) és C%(*) tömböket a főprogramban - feltéve, hogy az automatikus deklaráció nem teszi feleslegessé - deklarálni kell. C%(*) tömb méretének legalább max(A%,B%)-nak kell lenni.

A rutin az 1/2% segédváltozót használja.

A rutin a 11000 sorszámu utasitással kezdődik és a 11210 sorszámu utasitással fejeződik be.

```
11000 REM RUTIN KET NEMNEGATIV EGESZ SZAM KIVONASARA.
11010 REM AZ OPERANDUSZOKAT AZ A%(¥) ES B%(¥) TOMBOKBE KELL
      HELYEZNI, A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELE
      MEBE.
11020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK A%
      ILL. B% TOMBINDEXET IS.
11030 REM A KULONBSEG ABSZOLUT ERTEKE A C%(¥) TOMBBE KERUL.A
       GEP MEGADJA C%-OT IS,AMINEK JELENTESE AZ ELOZOKBOL ERT
      HETO.
11040 REM NEGATIV EREDMENY ESETEN C$ TARTALMA '-' LESZ, KULO
      NBEN URES.
11050 REM A RUTIN AZ %% SEGEDVALTOZOT HASZNALJA.
11060 C$='' : C%=A%
11070 IF A%>B% THEN 11140
11080 IF A%<B% THEN C%=B% : C$='-' : GOTO 11140
11090 FOR %%=A% TO 0% STEP -1%
11100 IF A%(%%)>B%(%%) THEN 11140
11110 IF A%(%%)<B%(%%) THEN C$='-' : GOTO 11140
11120 NEXT %%
11130 C%=0% : C%(0%)=0% : RETURN
11140 FOR %%=0% TO C%
11150 IF C$='' THEN IF %%>B% THEN C%(%%)=A%(%%) ELSE C%(%%)=
      A%(%%)-B%(%%)
11160 IF C$='-' THEN IF %%>A% THEN C%(%%)=B%(%%) ELSE C%(%%)
      =B%(%%)-A%(%%)
11170 NEXT %%
```

```
11180 FOR %%=0% TO C% : IF C%(%%)<0% THEN C%(%%)=C%(%%)+10%
      : C%(%%+1%)=C%(%%+1%)-1%
11190 NEXT %%
11200 FOR %%=C% TO 0% STEP -1% : IF C%(%%)>0% THEN C%=%% : R
      ETURN
11210 NEXT %%
```

Két nemnegativ egész szám szorzása
/MULRUT/

Ez a rutin az A%(*), B%(*) tömbökbe helyezett, akárhány jegyü, nemnegativ egész számok szorzatát számitja ki, s a szorzatot a C%(*) tömbbe helyezi.

A rutin a szorzás megszokott algoritmusa alapján müködik. A B%(*) tömbben levő szám jegyeivel rendre megszorozzuk az A%(*) tömbben levő szám jegyeit, s a helyiértékeket és az átviteleket is figyelembe véve a részletszorzatokat a C%(*) tömb megfelelő elemeiben összegezzük.

A rutin a C%(*) A%+B%+1-nél nagyobb indexü elemeit nem változtatja meg.

A főprogramban az A%(*), B%(*) és C%(*) tömböket - feltéve, hogy az automatikus deklaráció nem teszi feleslegessé - deklarálni kell. A C%(*) méretének legalább A%+B%+1-nek kell lenni.

A rutin az 1/2%, ←%, →%, →0% segédváltozókat használja.

A rutin a 12000 sorszámu utasitással kezdődik, s a 12190 sorszámu utasitással fejeződik be.

```
12000 REM RUTIN KET, AKARHANY JEGYU, NEMNEGATIV, EGESZ SZAM
      SZORZASARA
12010 REM A TENYEZOKET AZ A%(*) ILL. B%(*) TOMBBE KELL HELYE
      ZNI, A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELEMEBE.

12020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK A%,
       B% TOMBINDEXET IS.
12030 REM A SZORZAT A C%(*) TOMBBE KERUL, S A RUTIN MEGADJA
      C% ERTEKET IS.
12040 REM A RUTIN AZ A%(*),B%(*) TOMBOKET VALAMINT A C%(*) T
      OMB A%+B%+1%-NAL NAGYOBB INDEXU ELEMEIT NEM VALTOZTATJA
       MEG.
```

```
12050 REM C%(%) MERETENEK LEGALABB A%+B%+1%-NEK KELL LENNI.
12060 REM A RUTIN AZ %%,←%,→%,→0% SEGEDVALTOZOKAT HASZNALJA.
12070 IF (A%=0% AND A%(0%)=0%) OR (B%=0% AND B%(0%)=0%) THEN
      C%=0% : C%(0%)=0% : RETURN
12080 FOR %%=0% TO A%+B%+1% : C%(%%)=0% : NEXT %%
12090 FOR %%=0% TO B%
12100 →0%=0%
12110 FOR ←%=0% TO A%
12120 →%=C%(%%+←%)+A%(←%)*B%(%%)+→0%
12130 C%(%%+←%)=→%-→%/10%*10%
12140 →0%=→%/10%
12150 NEXT ←%
12160 C%(%%+←%)=C%(%%+←%)+→0%
12170 NEXT %%
12180 C%=A%+B% : IF C%(C%+1%)>0% THEN C%=C%+1%
12190 RETURN
```

Nemnegativ egész számok szorzása /II;/
/MULRUT2/

A szorzásra egy további szubrutint is készitünk, amely csak a tényezők jegyeinek a számára tett korlátozó feltétel teljesülése esetén müködik, ekkor azonban jobb az átlagos futási ideje.

A tényezőket most is az A%(*) ill. B%(*) tömbökbe kell helyezni, s a szorzat ugyancsak a C%(*) tömbbe kerül.

A rutin müködése azon alapszik, hogy egy kettős ciklus segitségével a C%(I) tömbelemben létrehozzuk a

$$\sum_{\substack{j+k=I \\ 0\le j\le A\% \\ 0\le k\le B\%}} A\%(j)\cdot B\%(k)$$

összeget, majd egy ujabb ciklusban az egyes helyiértékeken adódó átviteleket figyelembe véve a C%(*) elemeiben kialakitjuk a szorzat tényleges jegyeit.

A tényezők jegyeire tett megszoritás abból adódik, hogy a C%(*) tömb elemei egész tipusuak, igy a bennük tárolható legnagyobb szám 32767. Ha a kevesebb /nem több/ jegyü szám jegyeinek számát t-vel jelöljük, akkor a fenti algoritmus

szerint C%(*) egyik elemében sem képződhet t-nél több tagu összeg, ami maximálisan 81·t lehet. Az előző helyiértékről adódó, maximálisan 3276 átvitelt is figyelembe véve, t-re a következő elegendő feltétel adódik:

$$81 \cdot t + 3276 \leq 32767.$$

Mivel t egész, ebből $t \leq \frac{29491}{81} = 364.$

Ez azt jelenti, hogy ez a rutin használható, ha legalább az egyik tényező 365-nél kevesebb jegyü.

Az A%(*), B%(*), C%(*) tömbök deklarálására az előző rutinnál leirtak érvényesek.

A rutin az I/2%, ←%, →% segédváltozókat használja.

Ez a rutin is a 12000 sorszámu utasitással kezdődik, utolsó utasitásának sorszáma 12180.

```
12000 REM RUTIN KET NEMNEGATIV EGESZ SZAM SZORZASARA.
12010 REM A TENYEZOKET AZ A%(¥) ILL. B%(¥) TOMBOKBE KELL HEL
      YEZNI, A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELEMEB
      E.
12020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK A%
      ILL. B% TOMBINDEXET IS.
12030 REM A SZORZAT A C%(¥) TOMBBE KERUL. A GEP C%-OT IS MEG
      ADJA, AMINEK JELENTESE AZ ELOZOKBOL ERTHETO.
12040 REM A RUTIN A ½% , ←% ES →% SEGEDVALTOZOKAT HASZNALJA.

12050 IF (A%=0% AND A%(0%)=0%) OR (B%=0% AND B%(0%)=0%) THEN
      C%=0% : C%(0%)=0% : RETURN
12060 FOR ½%=0% TO A%+B%+1% : C%(½%)=0% : NEXT ½%
12070 FOR ½%=0% TO A%
12080 FOR ←%=0% TO B%
12090 C%(½%+←%)=C%(½%+←%)+A%(½%)¥B%(←%)
12100 NEXT ←%
12110 NEXT ½%
12120 FOR ½%=0% TO A%+B%
12130 →%=C%(½%)
12140 IF →%>9% THEN C%(½%+1%)=C%(½%+1%)+→%/10% : C%(½%)=→%-→
      %/10%¥10%
12150 NEXT ½%
12160 C%=A%+B%
12170 IF →%>9% THEN C%=C%+1%
12180 RETURN
```

Maradékos osztás
/DIVRUT/

Az alább ismertetett rutin az A%(*) ill. B%(*) tömbökben elhelyezett A ill. B, akárhány jegyü, nemnegativ egész számokon végez maradékos /euklideszi/ osztást, s a C hányadost a C%(*), az F maradékot az F%(*) tömbbe helyezi.

A rutin először megvizsgálja, hogy az osztó zérus-e, s ha igen, akkor hiba- és hangjelzést adva megszakitja a program futását.

Ha az osztó nem 0, akkor azt vizsgálja meg, hogy az osztandó kevesebb jegyü-e, mint az osztó, ekkor ugyanis a hányados 0, a maradék pedig éppen A.

A többi esetben a rutin az osztás közismert algoritmusát követi, melynek lényege, hogy a feladatot bizonyos számu /A%-B%+1/ olyan maradékos osztásra vezeti vissza, amely esetben a hányados értéke maximálisan 9 lehet, s e ciklusban végzett osztások hányadosai rendre megadják a keresett C hányados jegyeit, F pedig az utolsó ciklusléptésben végzett osztás maradéka.

Az algoritmusból ismeretes, hogy az egyes ciklus lépések során az osztandó eggyel több jegyü is lehet, mint az osztó. Ilyen esetben a ciklusléptés osztandójának első jegye által képviselt értéket az eggyel alacsonyabb helyiértéken vesszük figyelembe ugy, hogy az első jegy tizszeresét hozzáadjuk az eggyel alacsonyabb helyiértékü jegyhez. Hogy az első osztás is beilleszthető legyen a ciklusba, az F%(*) tömbbe már áthelyezett A osztandó elé helyezzünk egy 0 számjegyet. /F%(A%+1%) = 0/

Az egyes ciklusléptésekben szereplő osztásokat egy belső ciklus segitségével oldjuk meg ugy, hogy az osztandóból, - amely a mindenkori F%(*) tömbnek egy szelete - annyiszor vonjuk ki egymás után az osztót, ahányszor csak lehet anélkül, hogy a különbség negativ lenne. A lehetséges kivonások száma adja a C hányados megfelelő jegyét.

A külső ciklus A% ≥ B% esetén A%-B%+1 lépés után véget ér, s az utolsó ciklusiépés maradéka, ami pontosan a keresett F maradék, éppen az F%(*) tömbben van.

A C hányados jegyeinek a száma aszerint A%-B% ill. A%--B%+1, hogy a külső ciklus első lépése során kapott hányados 0-e, vagy sem. C% értéke igy megadható. C = 0 esetén C% = 0.

A maradék legmagasabb helyiértékü értékes jegyét egy ujabb ciklusban állapitjuk meg, igy F% is megadható.

Az A%(*), B%(*), C%(*), F%(*) tömböket a főprogramban deklarálni kell, ha nincs megfelelő automatikus deklaráció. C%(*) méretének legalább max(0,A%-B%+1)-nek, F%(*) méretének pedig legalább A%+1-nek kell lenni.

A rutin az 1/2%, ←%, ←0%, →%, →0% segédváltozókat használja.

A rutin a 13000 sorszámu utasitással kezdődik és a 13280 sorszámu utasitásig tart.

```
13000 REM RUTIN NEMNEGATIV EGESZ SZAM POZITIV EGESZ SZAMMAL
      VALO MARADEKOS OSZTASARA.
13010 REM AZ OPERANDUSZOKAT AZ A%(*) ES B%(*) TOMBOKBE KELL
      HELYEZNI, A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELE
      MEBE.
13020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK A%
      ILL. B% TOMBINDEXET IS.
13030 REM A HANYADOS A C%(*), A MARADEK AZ F%(*) TOMBBE KERU
      L, S A GEP - ERTELEMSZERUEN - C% ILL. F% ERTEKET IS MEG
      ADJA.
13040 REM A RUTIN AZ ½% , ←% , ←0% , →% ES →0% SEGEDVALTOZOK
      AT HASZNALJA.
13050 IF B%=0% AND B%(0%)=0% THEN ; "NULLAVAL OSZTAS!" : OUT
       6,255 : STOP
13060 C%=A%-B%
13070 FOR ½%=0% TO A% : F%(½%)=A%(½%) : NEXT ½% : F%=A%
13080 IF C%<0% THEN C%=0% : C%(0%)=0% : RETURN
13090 F%(A%+1%)=0%
13100 FOR ½%=0% TO C% : F%(F%-½%)=F%(F%-½%)+F%(F%-½%+1%)*10%

13110 ←0%=0%
13120 FOR ←%=B% TO 0% STEP -1%
13130 IF F%(C%+←%-½%)>B%(←%) THEN 13160
13140 IF F%(C%+←%-½%)<B%(←%) THEN 13220
13150 NEXT ←%
```

```
13160 ←0%=←0%+1% : →%=0% : →0%=0%
13170 FOR ←%=0% TO B%
13180 →0%=F%(C%+←%-½%)-B%(←%)-→%
13190 IF →0%<0% THEN F%(C%+←%-½%)=→0%+10% : →%=1% ELSE F%(C%
      +←%-½%)=→0% : →%=0%
13200 NEXT ←%
13210 GOTO 13120
13220 C%(C%-½%)=←0%
13230 NEXT ½%
13240 IF C%(C%)=0% AND C%>0% THEN C%=C%-1%
13250 FOR F%=B% TO 0% STEP -1% : IF F%(F%)>0% THEN RETURN
13260 NEXT F%
13270 F%=0%
13280 RETURN
```

Hatványozás a nemnegativ egész számok körében
/HAVRUT/

A következőkben leirt rutin az A%(*) tömbbe helyezett A nemnegativ egész számnak az X% változóba helyezett I nemnegativ egész kitevőre hatványozását végzi el, s a hatvány értékét a C%(*) tömbbe helyezi. Az alap elvileg akárhány jegyü szám, a kitevő maximálisan 32767 lehet. /Ez a korlátozás abból adódik, hogy X% egész tipusu változó./

A rutin müködését a mellékelt folyamatábrán is szemléltetjük.

A rutin először megvizsgálja, hogy az alap és a kitevő egyszerre 0-e, s ha igen, akkor hang- és hibajelzést adva megszakitja a program futását. Ha ez nem következik be, akkor azt vizsgálja, hogy az alap 0-e, s ha igen /a kitevő mostmár nem lehet 0/, akkor a hatvány értéke 0, s a szubrutin befejezi müködését. Ha ez sem teljesül, akkor megvizsgálja, hogy az A = 1 VAGY X = 0 logikai kifejezés értéke igaze, s ez esetben a hatvány értéke 1, s a rutin befejezi müködését.

Amennyiben az előző esetek egyike sem következik be, akkor az A alapot az ←2(*) tömbbe mentjük át, az X kitevőt pedig az ←4% munkaváltozóba töltjük, melynek mindenkori tartalmát a továbbiakban /a blokkdiagramon is/ egyszerüség ked-

véért Z-vel jelöljük. A számolás során keletkező részletszorzatok tárolására az ←1(*) tömböt használjuk, s kezdőértékként egyet helyezünk bele. E tömb aktuális tartalmát a továbbiakban P-vel jelöljük.

A rutin müködésének további részletezése céljából irjuk fel a hatvány kitevőjét a kettes számrendszerben:

$$Z = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 2^n + a_o$$

Az a_i értékek /O≤i≤n/ a Z szám 2 alapu számrendszerbeli jegyei, értéküK O vagy 1 lehet, és $a_n = 1$.
A számitandó hatvány ekkor igy irható:

$$A^Z = (A^{2^n})^{a_n} \cdot (A^{2^{n-1}})^{a_{n-1}} \cdot \ldots \cdot (A^2)^{a_1} \cdot A^{a_o} .$$

Látható, hogy a tényezők a_i kitevőjü hatványok, és az a_i kitevőjü hatvány alapja éppen az a_{i-1} kitevőjü hatvány alapjának a négyzete:

$$A^{2^i} = (A^{2^{i-1}})^2$$

Ebből adódik a következő algoritmus. Képezzük egy ciklusban az A szám 1, 2, 2^2, ..., 2^i, ..., 2^n kitevőjü hatványait, s amennyiben a 2^i kitevőnek megfelelő a_i jegy 1, akkor A^{2^i}-nel szorozzuk a már addig létrejött részletszorzatot,P-t. A Z szám kettes számrendszerbeli jegyeit, a_i-ket, 2-vel való maradékos osztások szorzatával határozzuk meg, s az adódó hányadosokat mindig ujra Z-vel jelöljük. Ha már eljutottunk a legmagasabb helyiértékü jegyhez /Z = a_n = 1/, akkor az eljárás véget ér, s az A lapnak az A%(*) tömbbe való visszatöltése után a rutin müködése befejeződik.

A rutin a hatvány alapját és kitevőjét nem változtatja meg, de általában megváltoztatja az A%(*) tömb A%-nál nagyobb indexü elemeit! /Az A%(*) tömböt ugyanis a hivott szorzórutin is használja./

A rutin az egyes ciklusiépések során vagy egyszer, vagy kétszer hivja a szorzórutint, egyrészt A emlitett hatványainak számitása, másrészt P-nek e hatványokkal való esetle-

ges szorzása céljából. Az A alap szóbanforgó hatványait a rutin ismételt négyzetreemelésekkel /önmagával való szorzásokkal/ számitja.

A főprogramban az A%(*), ←1%(*), (%(*),←2%(*), valamint a szorzórutinban szereplő B%(*) tömböt deklarálni kell, ha az az automatikus deklaráció miatt nem felesleges. A tömbök deklarálásánál figyelembe kell venni, hogy a rutin futása során A%(*)-ba kerülő maximális szám A^{2^n}, a B%(*) és ←1%(*)-be pedig maximálisan A^{2^n-1} kerülhet. A C%(*) tömbben el kell férnie a keresett hatványnak, de a szorzórutin tulajdonsága miatt ennél eggyel nagyobb méretre is szükség lehet.

A ←2%(*) tömb méretének legalább A%-nak kell lenni.

A rutin az 1/2%, ←%, ←1%, ←1%(*), ←2%, ←2%(*), ←4% és a hivott szorzórutinban szereplő segédváltozókat - beleértve a B% és B%(*)-ot is - használja.

A rutin müködéséhez szükség van valamelyik szorzórutin betöltésére, de annak kiválasztásánál figyelemmel kell lenni a MULRUT2-ben levő korlátozásra.

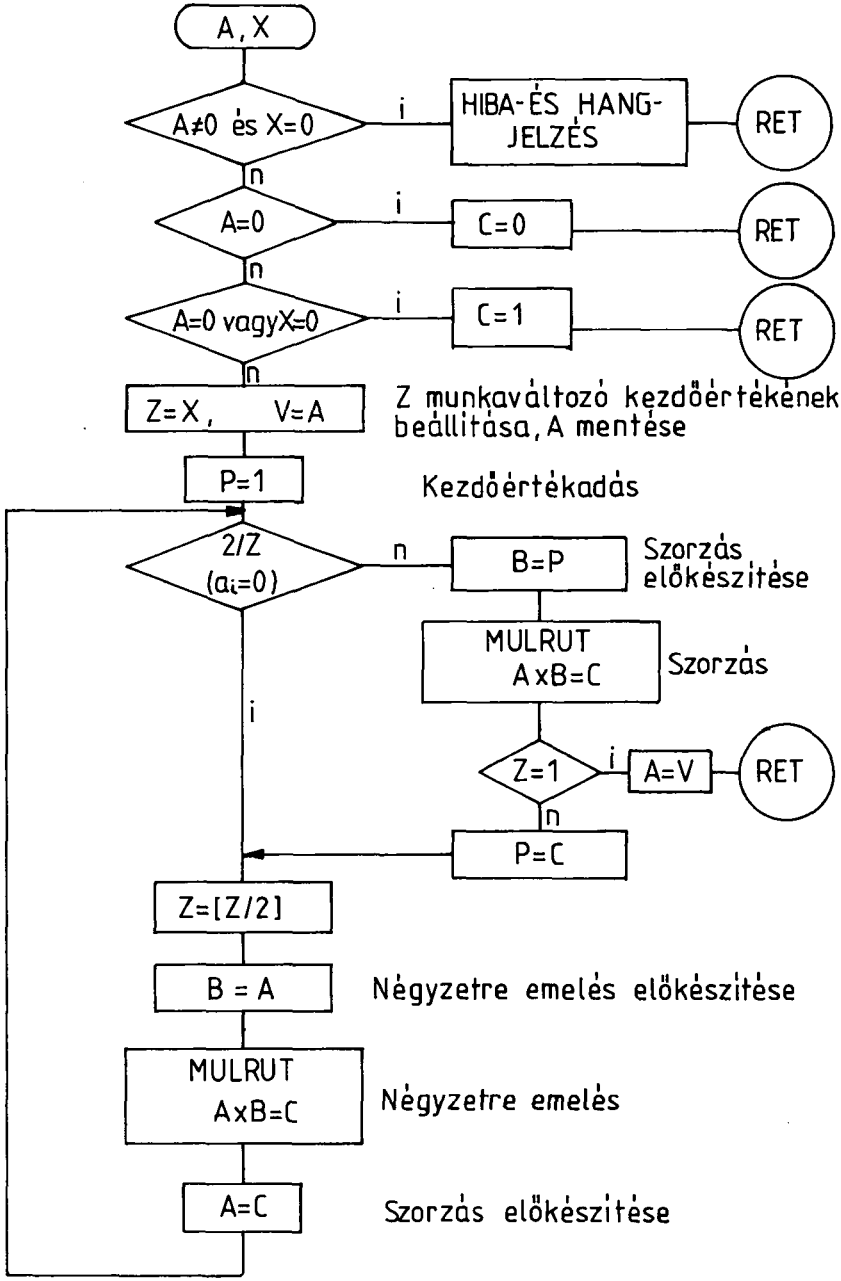
A rutin a 14000 sorszámu utasitással kezdődik és a 14220 sorszámu utasitásig tart.

Megjegyzés: Ennek a rutinnak a futási ideje lényegesen jobb az egyébként kézenfekvő $A^n = A^{n-1}$. A ismételt szorzáson alapuló rutinénál. Ez utóbbi esetben a szám n-edik hatványának számitásához szükséges szorzások száma n-1, mit az itt leirt rutin esetében legfeljebb $2 \cdot [\log_2 n]+1$, sőt átlagosan ennél is kevesebb. Még ha a szükséges előkészitéseket figyelembe vesszük is, ez lényegesen jobb futási időt eredményez.

```
14000 REM RUTIN NEMNEGATIV EGESZ SZAM NEMNEGATIV E
      GESZ KITEVOJU HATVANYANAK SZAMITASARA.
14010 REM AZ ALAPOT AZ A%(*) TOMBBE KELLHELYEZNI,A
       10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELEM
      EBE.
14020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU
      JEGY A% TOMBINDEXET IS.
14030 REM A KITEVOT AZ X% EGESZ TIPUSU VALTOZOBA K
      ELL HELYEZNI, IGY MAXIMALIS ERTEKE 32767 LEHE
      T.
14040 REM AZ EREDMENY A C%(*) TOMBBE KERUL, S A GE
      P MEGADAJA C% ERTEKET IS.
14050 REM A RUTIN HIVJA A MULRUT (12000)RUTINT.
14060 REM A RUTIN A B%, B%(*) ES ←-DEL KEZDODO, VA
      LAMINT A HIVOTT RUTINBAN SZEREPLO SEGEDVALTOZ
      OKAT HASZNALJA.
14070 REM A B%(*),←1%(*) ES ←2%(*) TOMBOKET A FOPR
      OGRAMBAN MEGFELELO MERETURE DEKLARALNI KELL.

14080 IF A%=0% AND A%(0%)=0% AND X%=0% THEN ; 'A H
      ATVANYNAK NINCS ERTELME.' : OUT 6,255 : STOP

14090 IF A%=0% AND A%(0%)=0% THEN C%=0% : C%(0%)=0
      % : RETURN
14100 IF (A%=0% AND A%(0%)=1%) OR X%=0% THEN C%=0%
       : C%(0%)=1% : RETURN
14110 ←4%=X% : FOR %%=0% TO A% : ←2%(%%)=A%(%%) :
      NEXT %% : ←2%=A%
14120 ←1%=0% : ←1%(0%)=1%
14130 IF ←4%-←4%/2%*2%=0% THEN GOTO 14180
14140 FOR %%=0% TO ←1% : B%(%%)=←1%(%%) : NEXT %%
      : B%=←1%
14150 GOSUB 12000 : REM SZORZAS
14160 IF ←4%=1% THEN FOR %%=0% TO ←2% : A%(%%)=←2%
      (%%) : NEXT %% : A%=←2% : RETURN
14170 FOR %%=0% TO C% : ←1%(%%)=C%(%%) : NEXT %% :
       ←1%=C%
14180 ←4%=←4%/2%
14190 FOR %%=0% TO A% : B%(%%)=A%(%%) : NEXT %% :
      B%=A%
14200 GOSUB 12000 : REM SZORZAS
14210 FOR %%=0% TO C% : A%(%%)=C%(%%) : NEXT %% :
      A%=C%
14220 GOTO 14130
```


A, X
A≠0 és X=0
i
HIBA-ÉS HANG-JELZÉS
RET
n
A=0
i
C=0
RET
n
A=0 vagy X=0
i
C=1
RET
n
Z=X, V=A
Z munkaváltozó kezdőértékének beállítása, A mentése
P=1
Kezdőértékadás
2/Z (aᵢ=0)
n
B=P
Szorzás előkészítése
MULRUT AxB=C
Szorzás
i
Z=1
i
A=V
RET
n
P=C
Z=[Z/2]
B = A
Négyzetre emelés előkészítése
MULRUT AxB=C
Négyzetre emelés
A=C
Szorzás előkészítése

Két szám legnagyobb közös osztója
/LNKORUT/

A számelméletben különösen, de a matematika legkülönbözőbb témaköreiben is gyakran előforduló feladat két egész szám legnagyobb közös osztójának meghatározása.

Az alább leirt rutin az A%(*) ill. B%(*) tömbökbe helyezett, akárhány jegyü, nemnegativ egész számok legnagyobb közös osztóját számitja ki, s azt a C%(*) tömbbe helyezi.

Célszerüségi okokból megállapodunk abban, hogy ha mindkét szám 0, akkor legnagyobb közös osztójuk is legyen 0. Ez nem okoz félreértést, ugyanakkor jó hasznát vehetjük több szám legnagyobb közös osztójának meghatározása során.

A rutin müködését a mellékelt blokkdiagramon is szemléltetjük.

A számolást ciklusban végezzük, s a ciklus első lépéseként megvizsgáljuk, hogy a B%(*)-ben tárolt szám 0-e, s ha igen, akkor a két szám legnagyobb közös osztója az A%(*)-ban tárolt szám /akkor is, ha az is 0/, s ezzel a rutin befejezte müködését. Egyéb esetben a két számon maradékos osztást végzünk, s a hányadost az A%(*) tömbbe, a maradékot a B%(*) tömbbe töltjük és visszatérünk a ciklus első lépéseként emlitett vizsgálathoz. Ezzel éppen az euklideszi algoritmust valósitjuk meg, amely véges számu lépés után véget ér, s az utolsó nem 0 maradék adja a két szám legnagyobb közös osztóját. /Könnyen ellenőrizhető, hogy a rutin akkor is helyesen müködik, ha az első szám 0./

A rutin müködéséből látható, hogy futása során az A%(*) és B%(*) tömbök megváltozhatnak!

A rutin müködéséhez be kell tölteni a DIVRUT nevü, maradékos osztást végző rutint.

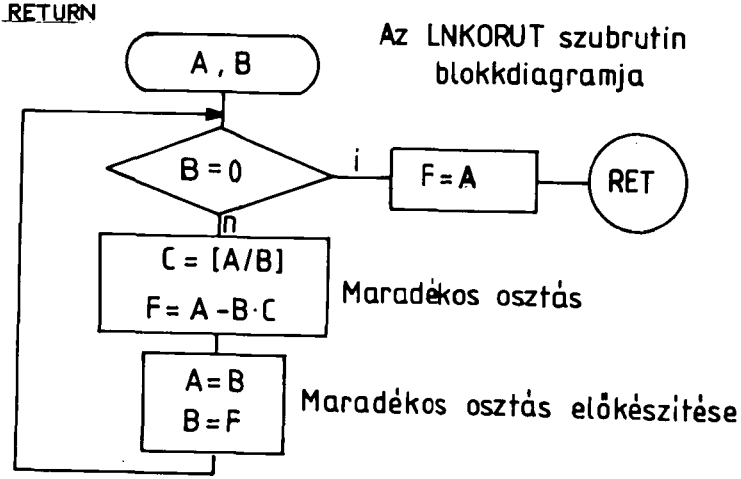
A főprogramban az A%(*), B%(*), C%(*) és F%(*) tömböket - hacsak az automatikus deklaráció nem teszi feleslegessé - deklarálni kell. C%(*)-ot elegendő max(A%,B%) méretüre deklarálni, s ekkora méretre szükség is lehet. F%(*)-ot pedig

max(A%+1,B%+1) méretüre elegendő deklarálni, bár esetenként ennél kisebb méret is megfelel.

A rutin csak a hivott rutinban is szereplő segédváltozókat használja.

A rutin 15000 sorszámu utasitással kezdődik és a 15120 sorszámu utasitásig tart.

```
15000 REM RUTIN KET NEMNEGATIV EGESZ SZAM LNKO-JAN
      AK SZAMITASARA.
15010 REM A KET SZAMOT AZ A%(¥) ILL.B%(¥) TOMBBE K
      ELL HELYEZNI, A 10^I HELYIERTEKU JEGYET A TOM
      B I INDEXU ELEMEBE.
15020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU
      JEGYEK A% ILL. B% TOMBINDEXET IS.
15030 REM AZ EREDMENY AZ F%(¥) TOMBBE KERUL,S F%-0
      T IS MEGADJA A GEP.HA MINDKET SZAM 0, AKKOR L
      NKO-JUK 0 LESZ.
15040 REM A RUTIN HIVJA A MARADEKOS OSZTAST VEGZO
      RUTINT ( DIVRUT :  13000-TOL).
15050 REM A RUTIN AZ ¥% , VALAMINT A HIVOTT RUTINB
      AN SZEREPLO SEGEDVALTOZOKAT HASZNALJA.
15060 IF B%=0% AND B%(0%)=0% THEN 15110
15070 GOSUB 13000 : REM OSZTAS
15080 FOR ¥%=0% TO B% : A%(¥%)=B%(¥%) : NEXT ¥% :
      A%=B%
15090 FOR ¥%=0% TO F% : B%(¥%)=F%(¥%) : NEXT ¥% :
      B%=F%
15100 GOTO 15060
15110 FOR ¥%=0% TO A% : F%(¥%)=A%(¥%) : NEXT ¥% :
      F%=A%
15120 RETURN
```



Két szám összehasonlitása
/COMPRUT/

Adatok gépi feldolgozása során igen gyakori feladat annak eldöntése, hogy két szám közül melyik a nagyobb, illetve hogy egyenlőek-e. Az ABC-80 tipusu számitógép ezt a beépitett COMP% függvény segitségével oldja meg. Ez a függvény maximálisan 29 karakteren ábrázolható valós számok esetén használható. Itt egy olyan rutint készitünk, amely csak a nemnegativ egész számok összehasonlitására alkalmas, viszont az értékes jegyek számára - a futási időn és memóriakapacitáson kivül - nincs korlátozás.

A rutin az A%(*) ill. B%(*) tömbökben tárolt A ill. B nemnegativ egész számokat hasonlitja össze, és az

$$S\% = \begin{cases} 1, & \text{ha } A > B \\ 0, & \text{ha } A = B \\ -1, & \text{ha } A < B \end{cases}$$

függvény értékét számitja ki. Ez összhangban van a COMP% beépitett függvénnyel.

Röviden leirjuk a rutin müködési elvét. Ha valamelyik számnak több jegye van /a megállapodás szerinti számábrázolásban csak értékes jegy szerepel!/, akkor az a nagyobb. Ha a két szám jegyeinek száma egyenlő, akkor a két szám jegyeit egy ciklusban, csökkenő helyiértékük sorrendjében összehasonlitjuk, s amelyikben előbb találunk nagyobb jegyet, az a nagyobb. Ha ilyen jegy nincs, akkor a két szám egyenlő.

A rutin az 1/2% segédváltozót használja. Első utasitása 21000, utolsó utasitása 21120 sorszámu.

```
21000 REM RUTIN AZ A%(*),B%(*) TOMBOKBEN MEGADOTT
      NEMNEGATIV EGESZ SZAMOK OSSZEHASONLITASARA.
21010 REM A 10^I HELLYIERTEKU JEGYET A TOMB I INDE
      XU ELEMEBE KELL HELYEZNI.
```

```
21020 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU
      JEGYEK A% ILL.B% TOMBINDEXET IS.
21030 REM A RUTIN AZ S%-BA ASZERINT HELYEZ  1-ET,
      0-T ILL.-1-ET,HOGY A KET SZAM KOZT A >, AZ =
      ILL. A < RELACIO IGAZ.
21040 REM A RUTIN AZ %% SEGEDVALTOZOT HASZNALJA.
21050 IF A%>B% THEN S%=1% : RETURN
21060 IF A%<B% THEN S%=-1% : RETURN
21070 FOR %%=A% TO 0% STEP -1%
21080 IF A%(%%)>B%(%%) THEN S%=1% : RETURN
21090 IF A%(%%)<B%(%%) THEN S%=-1% : RETURN
21100 NEXT %%
21110 S%=0%
21120 RETURN
```

Aritmetikai tömb decimális karakterlánccá alakitása /TOMKAR/

Programok összeállitása, adatok kényelmes tárolása és kiiratása céljából szükség lehet arra, hogy olyan aritmetikai tömböket, melyeknek elemei egész számok jegyei, decimális karakterlánccá alakitsunk. Erre a célra készült az alább ismertetett rutin.

A rutin a C%(*) ill. F%(*) aritmetikai tömbökben a szokásos módon ábrázolt nemnegativ egész számokat decimális karakterlánccá alakitja, s a C$ ill. F$ karakteres tipusu változóba tölti.

Müködése azon alapszik, hogy a C%(*) ill. F%(*) tömbök elemeit, tehát a szóban forgó egész számok számjegyeit - mint aritmetikai adatokat - egy ciklusban karakteres tipusuvá alakitja, s azokat konkatenálás utján a C$ ill. F$ karakteres változóba tömöriti.

Az átalakitás azon mulik, hogy a számjegyeknek, mint karaktereknek az ASCII kódjai rendre 48-cal nagyobbak, mint a jegyek alaki értéke, vagyis:

"i" = CHR$(i+48) $0 \le i \le 9$.

A rutin a C%(*) és F%(*) tömböket nem változtatja meg. Nem végzi el a rutin a C%(*) ill. F%(*) tömb átalakitását,

ha C1$ = "N" ill. F1$ = "N", s ekkor a C$ ill. F$ tartalma is változatlan marad.

A C$ és F$ változók hosszát a főprogramban legalább C%+1, ill. F%+1-re deklarálni kell, hacsak az automatikus deklaráció nem elegendő.

A rutin az 1/2% segédváltozót használja. Első utasitás 23000, utolsó utasitása pedig 23110 sorszámu.

```
23000 REM RUTIN A C%(%) ILL. F%(%) ARITMETIKAI TOM
      BBEN TAROLT NEMNEG EG. SZ. C$ ILL. F$ DEC. KA
      RAKTERLANCCA ALAKITASARA.
23010 REM A SZAMOK 10^I HELYIERTEKU JEGYET A MEGFE
      LELO TOMB I INDEXU ELEMEBE KELL HELYEZNI.
23020 REM MEG KELL ADNI A SZAMOK LEGMAGASABB HELYI
      ERTEKU JEGYENEK C% ILL. F% TOMBINDEXET IS.
23030 REM A RUTIN A C%(%) ILL. F%(%) TOMB ATALAKIT
      ASAT NEM VEGZI EL, HA C1$='N' ILL. F1$='N'
23040 REM EKKOR A C$ ILL F$ KARAKTERES VALTOZOK NE
      M VALTOZNAK MEG.
23050 REM A RUTIN AZ %% SEGEDVALTOZOT    HASZNALJA

23060 DIM C1$=1%,F1$=1%
23070 IF C1$='N' THEN 23090
23080 C$='' : FOR %%=C% TO 0% STEP -1% : C$=C$+CHR
      $(C%(%%)+48%) : NEXT %%
23090 IF F1$='N' THEN RETURN
23100 F$='' : FOR %%=F% TO 0% STEP -1% : F$=F$+CHR
      $(F%(%%)+48%) : NEXT %%
23110 RETURN
```

Decimális karakterlánc aritmetikai tömbbé alakitása /KARTOM/

A korábban már leirt szubrutinokat /alapmüveletek, hatványozás, hasonlitás, stb./ ugy készitettük, hogy bemenő adataikat - amelyek nemnegativ egész számok - aritmetikai tömbök formájában képesek fogadni, a bevezetőben már ismertetett módon. Gyakran kényelmesebb az adatok decimális karakterláncként való beolvasása, tárolása. Az is gyakori, hogy az adatok tőlünk függetlenül decimális karakterláncként vannak megadva.

Elkészitünk ezért egy szubrutint, amely a szóbanforgó szubrutinok hivása előtt az A$, B$ decimális karakterlánc tipusu adatokat az A%(*), B%(*) aritmetikai tömbökké alakitja, és megadja A% ill. B% értékét is. /Ügyeljünk arra, hogy A$ és B$ csak számjegyeket tartalmazhat!/

A rutin müködése a következő: A LEN beépitett függvény alkalmazásával megállapitja a karakterlánc hosszát, annak ismeretében egy ciklusban a MID$ beépitett függvény segitségével leválasztja a lánc egyes karaktereit, azokat az ugyancsak beépitett VAL függvény segitségével aritmetikai adattá alakitja, és a szóbanforgó tömb megfelelő indexü elemébe helyezi. Az A$ ill. B$ hosszából egyet-egyet levonva adódik A% ill. B%.

Ha az A$ vagy B$ karakterlánc valamelyike üres, akkor a rutin a neki megfelelő tömböt nem változtatja meg!

A főprogramban az A%(*), B%(*) tömböket - szem előtt tartva azok további felhasználását -, valamint az A$, B$ hosszát deklarálni kell, hacsak az automatikus deklarálás nem elegendő.

A rutin az 1/2% segédváltozót használja.

A rutin a 22000 sorszámu utasitástól a 22090 sorszámu utasitásig tart.

```
22000 REM RUTIN AZ A$ ILL. B$ DECIMALIS KARAKTERLA
      NCOK A%(*) ILL. B%(*) ARITMETIKAI TOMBOKKE AL
      AKITASARA.
22010 REM A 10^I HELYIERTEKU JEGYET A TOMB I INDEX
      U ELEMEBE HELYEZI.
22020 REM A GEP MEGADJA A LEGMAGASABB HELYIERTEKU
      JEGYEK A% ILL. B% TOMBINDEXET IS.
22030 REM HA VALAMELYIK KARAKTERES VALTOZO URES
      ,AKKOR A RUTIN A MEGFELELO TOMBOT NEM VALTOZT
      ATJA MEG.
22040 A%=LEN(A$)-1 : B%=LEN(B$)-1
22050 IF A%<0% THEN GOTO 22070
22060 FOR %%=0% TO A% : A%(%%)=VAL(MID$(A$,A%+1%-%
      %,1%)) : NEXT %%
22070 IF B%<0% THEN RETURN
22080 FOR %%=0% TO B% : B%(%%)=VAL(MID$(B$,B%+1%-%
      %,1%)) : NEXT %%
22090 RETURN
```

Ismétlés nélküli variációk száma
Faktoriális
/VARRUT/

A következőkben egy olyan szubrutint ismertetünk, amely az N%(*) tömbbe helyezett N ill. K%(*) tömbbe helyezett K, akárhány jegyü, nemnegativ egész számok esetén kiszámitja az N elemből képezhető K-ad osztályu ismétlés nélküli variációk V_N^K számát, s azt a C%(*) tömbbe tölti.

N = K esetén - mint ismeretes - $V_N^K = N!$. Ha a rutinnal N! értékét kivánjuk számittatni, akkor nem szükséges N-et a K%(*) tömbbe is betölteni, ha a rutin hivásakor F$ = "F". /Mindig ügyelni kell F$ aktuális értékére, F$ = "F" esetén ugyanis a rutin mindig N!-t számitja!/

A rutin a számolást az ismert

$$V_N^K = N\cdot(N-1)\cdot\ldots\cdot(N-(K-1))$$

képlet alapján ciklusban végzi, amit a mellékelt blokkdiagram is szemléltet.

Mielőtt a rutin müködését ismertetnénk, megállapodunk abban, hogy a nulla tényezős szorzaton 1-et, egy tényezős szorzaton pedig magát a számot értjük.

Az 1/2 7%(*) tömbben tároljuk az addig összeszorzott tényezők szorzatát, amit - rövidség kedvéért P-vel jelölünk, s kezdőértékét egynek választjuk. Az 1/2 6%(*) tömbben az addig már összeszorzott tényezők számát tároljuk, ezt T-vel jelüljük, s kezdőértékét O-nak választjuk.

Egy-egy ciklus a P számnak az N-T tényezővel való szorzását végzi el, ahol $O \leq T \leq K-1$. A ciklus elején a rutin megvizsgálja, hogy összeszoroztuk-e már a szükséges számu tényezőt, vagyis fennáll-e a $T \geq K$ egyenlőtlenség. Ha nem, akkor kiszámitjuk N-T értékét, s azzal megszorozzuk P-t. Utána T értékét megnöveljük 1-gyel, és visszatérünk az összeszorzott tényezők számának ellenőrzéséhez. Ha már összeszoroztuk a megfelelő számu tényezőt, akkor P-t áttöltjük a C%(*) tömbbe, és a rutin befejezi müködését.

Könnyen ellenőrizhető, hogy K = 0 esetén a ciklus egyszer sem fut le, és igy V_N^0-ra N-től függetlenül 1 adódik.

K > N esetén a rutin V_N^K-ra 0-t ad, összhangban V_N^K jelentésével, de ekkor feleslegesen sok számolást végez, és feleslegesen nagy méretü tömbökre van szükség. Célszerü tehát a rutint csak $0 \leq K \leq N$ esetén használni.

A főprogramban az A%(*), B%(*), C%(*), 1/2 6%(*), 1/2 7%(*) tömböket - ha az automatikus deklaráció nem elegendő - deklarálni kell. Az 1/2 7%(*) és C%(*) tömbben el kell férnie a végeredménynek, és K > N esetén N!-nak is. A C%(*) tömb méretét az igy adódó értéknél legalább eggyel nagyobbra kell választani /a szorzórutin tulajdonsága miatt/. A%(*) tömbben el kell férnie $K \leq N$ esetén V_N^{K-1}-nek, K > N esetén N!-nak is. Az 1/2 6%(*) tömbben el kell férnie K+1-nek, a B%(*) tömbben pedig N és K közül a nagyobbiknak.

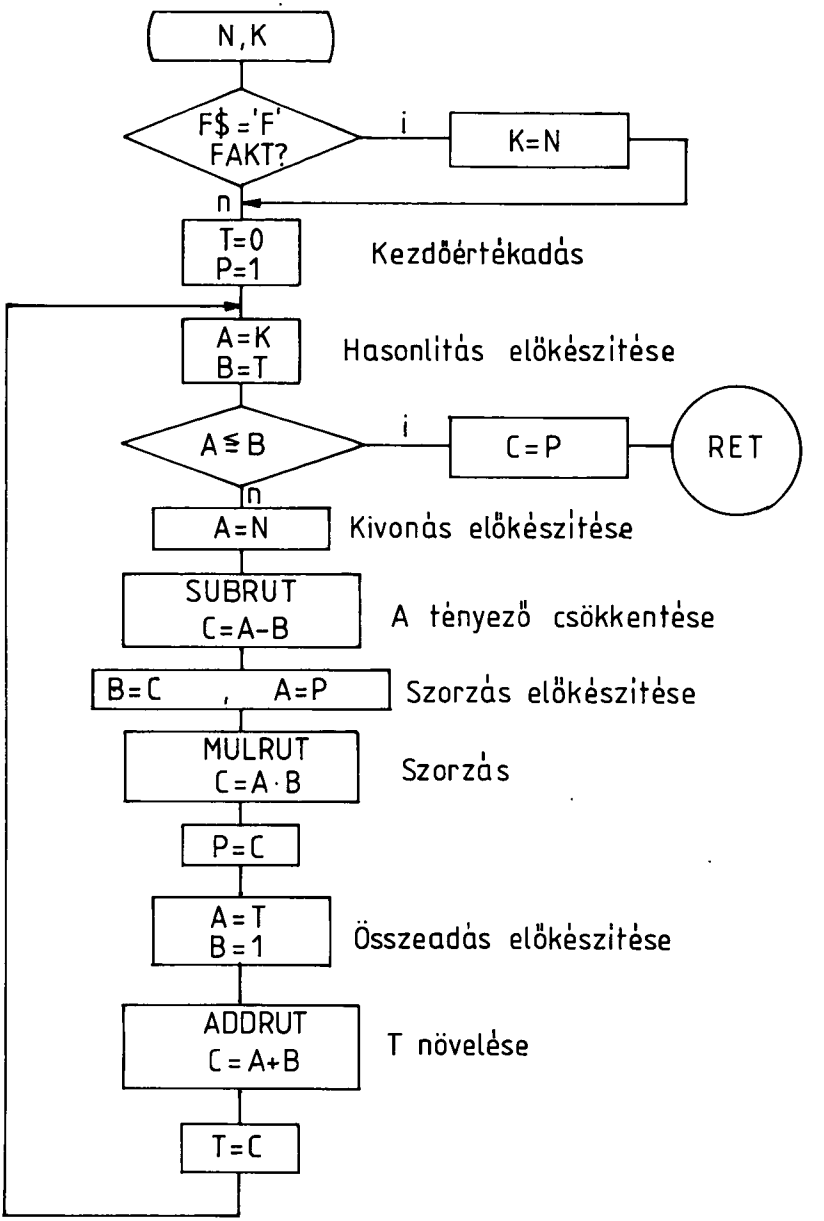
A rutin hivja az ADDRUT, SUBRUT, MULRUT vagy MULRUT2 és COMPRUT szubrutinokat, futtatáskor tehát ezeket is be kell tölteni a gépbe.

A rutin az 1/2 6%, 1/2 6%(*), 1/2 7%, 1/2 7%(*), valamint a hivott rutinokban szereplő segédváltozókat használja. Gondoljunk arra is, hogy a SUBRUT és COMPRUT rutinok hivása miatt a C$ és S% változók értéke is megváltozhat!

A rutin a 17000 sorszámu utasitással kezdődik, és a 17270 sorszámuig tart.

```
17000 REM RUTIN AZ N ELEMBOL KEPEZHETO K-AD OSZTALYU ISMETLES NELKULI VARIACIOK SZAMANAK MEGHATAROZASARA.
17010 REM AZ N ES K NEMNEGATIV EGESZ SZAMOKAT AZ N%(*) ILL. K%(*) TOMBBE KELL HELYEZNI.
17020 REM A 10^I HELYIERTEKU JEGYET A TOMB I INDEXU ELEMEBE KELL TENNI.
17030 REM MEG KELL ADNI A LEGMAGASABB HELYIERTEKU JEGYEK N% ILL. K% TOMBINDEXET IS.
17040 REM A PROGRAM SPECIALIS ESETKENT N FAKTORIAL IS ERTEKET IS KISZAMITJA.
17050 REM HA F$='F', AKKOR A RUTIN N FAKTORIALIST SZAMITJA, S K ERTEKET NEM KELL MEGADNI.
17060 REM AZ EREDMENY A C%(*) TOMBBE KERUL,S C%-OT
```

```
      IS MEGADJA A RUTIN.
17070 REM A RUTIN AZ ADDRUT (10000), MULRUT (13000
      ), SUBRUT(11000) ES COMPRUT (21000) SZUBRUTIN
      OKAT HIVJA.
17080 REM A RUTIN AZ A%(*),A%,B%(*),B%,C%(*),C%,S%
      ,%6%(*),%6%,%7%(*),%7% ,ES A HIVOTT RUTINOK S
      EGEDVALTOZOIT HASZNALJA.
17090 REM A TOMBOKET A FOPROGRAMBAN MEGFELELO MERE
      TURE DEKLARALNI KELL!
17100 IF F$='F' THEN FOR %%=0% TO N% : K%(%%)=N%(%
      %) : NEXT %% : K%=N%
17110 %6%=0% : %6%(0%)=0% : %7%=0% : %7%(0%)=1%
17120 FOR %%=0% TO K% : A%(%%)=K%(%%) : NEXT %% :
      A%=K%
17130 FOR %%=0% TO %6% : B%(%%)=%6%(%%) : NEXT %%
      : B%=%6%
17140 GOSUB 21000 : REM HASONLITAS
17150 IF S%<1% THEN FOR %%=0% TO %7% : C%(%%)=%7%(
      %%) : NEXT %% : C%=%7% : RETURN
17160 FOR %%=0% TO N% : A%(%%)=N%(%%) : NEXT %% :
      A%=N%
17170 FOR %%=0% TO %6% : B%(%%)=%6%(%%) : NEXT %%
      : B%=%6%
17180 GOSUB 11000 : REM KIVONAS
17190 FOR %%=0% TO C% : B%(%%)=C%(%%) : NEXT %% :
      B%=C%
17200 FOR %%=0% TO %7% : A%(%%)=%7%(%%) : NEXT %%
      : A%=%7%
17210 GOSUB 12000 : REM SZORZAS
17220 FOR %%=0% TO C% : %7%(%%)=C%(%%) : NEXT %% :
      %7%=C%
17230 FOR %%=0% TO %6% : A%(%%)=%6%(%%) : NEXT %%
      : A%=%6%
17240 B%=0% : B%(0%)=1%
17250 GOSUB 10000 : REM OSSZEADAS
17260 FOR %%=0% TO C% : %6%(%%)=C%(%%) : NEXT %% :
      %6%=C%
17270 GOTO 17120
```


N,K
F$ ='F'
FAKT?
i
K=N
n
T=0
P=1
Kezdőértékadás
A=K
B=T
Hasonlítás elökészítése
A≦B
i
C=P
RET
n
A=N
Kivonás elökészítése
SUBRUT
C=A-B
A tényező csökkentése
B=C , A=P
Szorzás elökészítése
MULRUT
C=A·B
Szorzás
P=C
A=T
B=1
Összeadás elökészítése
ADDRUT
C=A+B
T növelése
T=C

Binomiális együtthatók számitása
/BINRUT/

Az általános és középiskolai oktatásban fontos szerepet töltenek be a halmazok, valamint a részhalmazaikból képezhető véges sorozatok. Idevágó alapvető fontosságu feladat az N elemből képezhető K-ad osztályu ismétlés nélküli kombinációk C_N^K számának meghatározása. Szokásos a $C_N^K = \binom{N}{K}$ jelölése és az $(a+b)^N$ hatvánnyal való kapcsolatuk miatt a binomiális együttható elnevezés. Ismert, hogy $\binom{N}{K}$ egyben az N elemből képezhető ismétléses permutációk száma is, ha közülük K ill. N-K elem egymástól nem különböztethető meg. Ugyancsak megfelelő binomiális együttható adja az N elemből képezhető K-ad osztályu ismétléses kombinációk számát is: $\binom{N+K-1}{K}$.

Az ilyen jellegü feladatok megoldása során viszonylag kis kiinduló adatok esetén is rendkivül nagy rész- ill. végeredmények adódhatnak. Időnként hasznos lehet ezek pontos, tényleges előállitása is.

Erre a célra készült az alábbi szubrutin, amely az N%(*) tömbbe helyezett N ill. a K%(*) tömbbe helyezett K, akárhány jegyü, nemnegativ egész számok esetén kiszámitja $\binom{N}{K}$értékét és azt a C%(*) tömbbe helyezi.

A rutin részeredményként kiszámitja az összes $\binom{N}{T}$, $0 \leq T \leq K$ binomiális együtthatót is, és a felhasználó kivánságára - ha a rutin hivásakor I$ = "I" - azokat kiirja a képernyőre.

A kitüzött feladatot a már leirt VARRUT nevü rutinnal is meg tudnánk oldani: kiszámitanánk vele a V_N^K-t, majd K!-t és a kettő hányadosa adná az $\binom{N}{K}$ binomiális együtthatót. Igy azonban a szükségesnél általában sokkal nagyobb számokkal kellene számolnunk, ami egyrészt a futási időt rontaná, másrészt a számolás nagyobb méretü tömböket igényelne. Ezért itt a számolást az

$$\binom{N}{K} = \prod_{T=1}^{K} \frac{N-(T-1)}{T}$$

összefüggés alapján, T egyes értékeinek megfelelő ciklusban végezzük, amit a mellékelt blokkdiagramon is szemléltetünk.

A számolás ilyen szervezésének több előnye is van: az egyes ciklusok befejezésekor részletszorzatként éppen $\binom{N}{T}$, $0 < T \leq K$ értékeit kapjuk, amelyek nyilván egész számok, s ez egyben azt is jelenti, hogy a számolás során végig az egész számok körében maradunk.

A rutin az 1/2 1%(*) tömböt a P-vel jelölt részletszorzatok tárolására, az 1/2 2%(*) tömböt pedig a T-vel jelölt "ciklusváltozó" tárolására használja és mindkettő kezdőértékét 1-nek választjuk. Az 1/2 0%(*) tömbben tárolt szám mindenkori értékét V-vel jelöljük, s kezdőértékét N-re állitjuk.

A ciklus egy-egy lépése a P részletszorzat V = N-(T-1)-gyel való szorzását, majd a szorzat T-vel való osztását végzi - I = "I" esetén P-t kiirja a képernyőre - és előkészíti a következő lépést, vagyis T értékét 1-gyel növeli, V értékét pedig eggyel csökkenti. A rutin az egyes ciklusok kezdetén megvizsgálja, hogy összeszoroztuk-e már a megfelelő számu tényezőt, vagyis hogy fennáll-e a T > K reláció, s ha igen, akkor - miután P értékét a C%(*) tömbbe töltötte - befejezi müködését, különben folytatódik a ciklus végrehajtása.

Könnyen ellenőrizhető, hogy K = 0 esetén a ciklus egyszer sem fut le, s igy a P = 1 kezdőértékadás folytán $\binom{N}{0}$-ra, N-től függetlenül mindig 1 adódik, $\binom{N}{0}$ értelmezésének megfelelően.

A rutin K > N esetén is müködik, és ilyenkor $\binom{N}{K}$-ra 0-t ad, ami összhangban van $\binom{N}{K}$ általánosabb értelmezésével. Ekkor azonban feleslegesen sok számolást végez és a szükségesnél nagyobb méretü tömböket igényel, nem célszerü használni.

Megjegyezzük, hogy ha csak az $\binom{N}{K}$ binomiális együtthatóra van szükségünk, akkor $K > \frac{N}{2}$ esetén K helyett N-K-val célszerü számolni, ugyanis $\binom{N}{K} = \binom{N}{N-K}$.

A rutin az N%(*) és K%(*) tömböket nem változtatja meg, de a C$, S% és F%(*) értékét megváltoztathatja a SUBRUT, COMPRUT és DIVRUT rutinok hivása folytán.

A rutin az 1/2%, 1/2 O%, 1/2 O%(*), 1/2 1%(*), 1/2 2%, 1/2 2%(*), valamint a hivott - ADDRUT, SUBRUT, MULRUT vagy MULRUT2, DIVRUT és COMPRUT - rutinokban szereplő segédváltozókat használja. Az itt felsorolt rutinokat a futtatáshoz be kell tölteni!

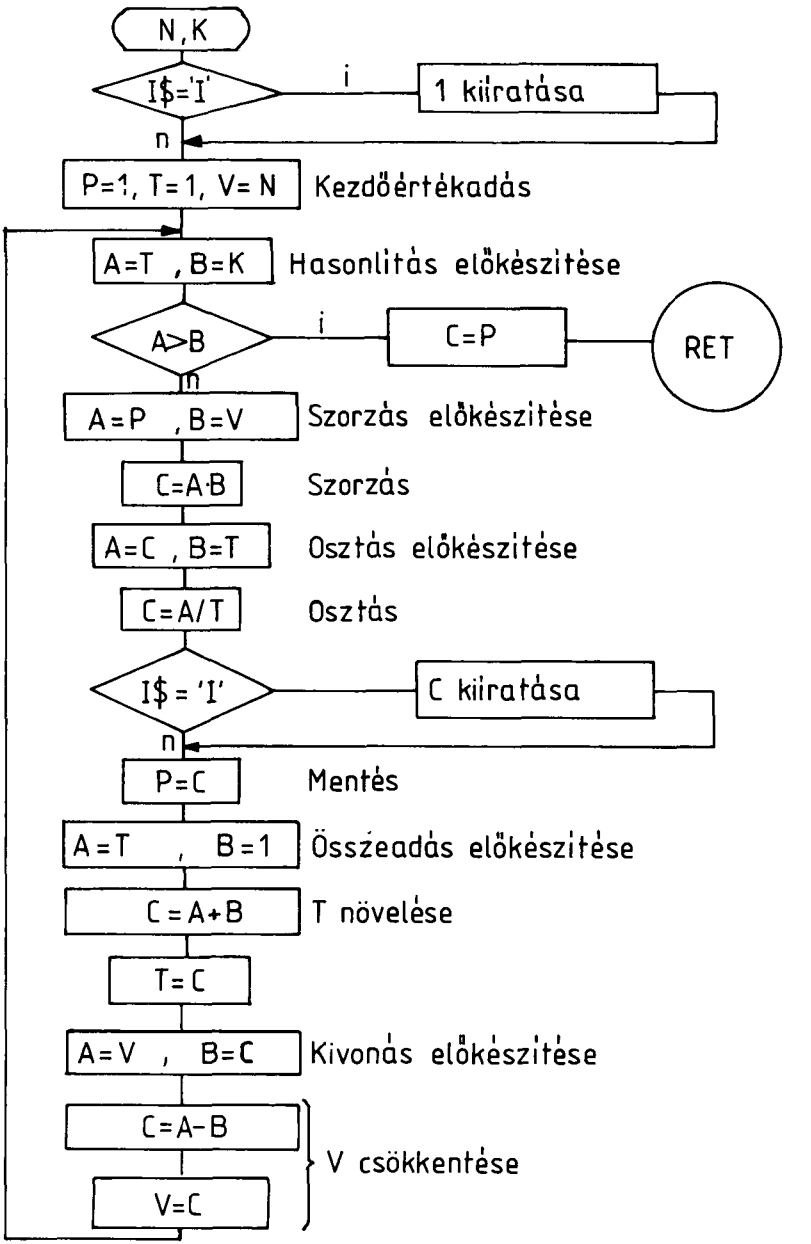
A főprogramban az 1/2 O%(*), 1/2 1%(*), 1/2 2%(*), A%(*), B%(*), C%(*) és F%(*) tömböket - ha az automatikus deklaráció nem elegendő - deklarálni kell. A szükséges méreteket itt csak K N esetére foglaljuk össze. A B%(*), 1/2 O%(*) és 1/2 2%(*) tömböket elegendő N%+1 méretüre deklarálni. Az A%(*), C%(*), 1/2 1%(*) és F%(*) tömbökben $K \leq \frac{N}{2}$ esetén el kell férnie a végeredmény K-szorosának, $K > \frac{N}{2}$ esetén az N-hez tartozó legnagyobb binomiális együttható,

$$\left(\binom{N}{\left[\frac{N}{2}\right]}\right) \left[\frac{N+1}{2}\right]$$ -szeresének kell elférnie bennük.

C%(*) és F%(*) méretének az ehhez szükségesnél legalább egygyel nagyobbnak kell lenni a szorzó ill. osztórutin tulajdonsága miatt.

A rutin a 18000 sorszámu utasitással kezdődik, s a 18300 sorszámuig tart.

```
18030 REM AZ EREDMENY A C%(¥) TOMBBE KERUL, S A GE
      P MEGADJA C% ERTEKET IS.
18040 REM HA I$='I', AKKOR A RUTIN KIIRJA N ALATT
      I ERTEKEIT K-NAL NEM NAGYOBB, NEMNEGATIV I ER
      TEKEKRE.
18050 REM A RUTIN HIVJA AZ ADDRUT (10000), SUBRUT
      (11000), MULRUT (12000), DIVRUT (13000) ES CO
      MPRUT (21000) RUTINOKAT.
18060 REM A RUTIN AZ A%,A%(¥),B%,B%(¥),¾%,¾0%,¾0%(
      ¥),¾1%,¾1%(¥),¾2%,¾2%(¥) ES A HIVOTT RUTINOK
      SEGEDVALTOZOIT HASZNALJA
18070 REM A TOMBOKET A FOPROGRAMBAN MEGFELELO MERE
      TURE DEKLARALNI KELL!
18080 IF I$='I' THEN ; '1'
18090 FOR ¾%=0% TO N% : ¾0%(¾%)=N%(¾%) : NEXT ¾% :
      ¾0%=N%
18100 ¾1%=0% : ¾1%(0%)=1% : ¾2%=0% : ¾2%(0%)=1%
18110 FOR ¾%=0% TO ¾2% : A%(¾%)=¾2%(¾%) : NEXT ¾%
      : A%=¾2%
18120 FOR ¾%=0% TO K% : B%(¾%)=K%(¾%) : NEXT ¾% :
      B%=K%
18130 GOSUB 21000 : REM HASONLITAS
18140 IF S%=1% THEN FOR ¾%=0 TO ¾1% : C%(¾%)=¾1%(¾
      %) : NEXT ¾% : C%=¾1% : RETURN
18150 FOR ¾%=0% TO ¾1% : A%(¾%)=¾1%(¾%) : NEXT ¾%
      : A%=¾1%
18160 FOR ¾%=0% TO ¾0% : B%(¾%)=¾0%(¾%) : NEXT ¾%
      : B%=¾0%
18170 GOSUB 12000 : REM SZORZAS
18180 FOR ¾%=0% TO C% : A%(¾%)=C%(¾%) : NEXT ¾% :
      A%=C%
18190 FOR ¾%=0% TO ¾2% : B%(¾%)=¾2%(¾%) : NEXT ¾%
      : B%=¾2%
18200 GOSUB 13000 : REM OSZTAS
18210 IF I$='I' THEN FOR ¾%=C% TO 0% STEP -1% : ;
      CHR$(C%(¾%)+48%); : NEXT ¾% : ;
18220 FOR ¾%=0% TO C% : ¾1%(¾%)=C%(¾%) : NEXT ¾% :
      ¾1%=C%
18230 FOR ¾%=0% TO ¾2% : A%(¾%)=¾2%(¾%) : NEXT ¾%
      : A%=¾2%
18240 B%=0% : B%(0%)=1%
18250 GOSUB 10000 : REM OSSZEADAS
18260 FOR ¾%=0% TO C% : ¾2%(¾%)=C%(¾%) : NEXT ¾% :
      ¾2%=C%
18270 FOR ¾%=0% TO ¾0% : A%(¾%)=¾0%(¾%) : NEXT ¾%
      : A%=¾0%
18280 GOSUB 11000 : REM KIVONAS
18290 FOR ¾%=0% TO C% : ¾0%(¾%)=C%(¾%) : NEXT ¾% :
      ¾0%=C%
18300 GOTO 18110
```


N,K
I$='I'
i
1 kiiratása
n
P=1, T=1, V=N
Kezdőértékadás
A=T , B=K
Hasonlitás előkészitése
A>B
i
C=P
RET
n
A=P , B=V
Szorzás előkészitése
C=A·B
Szorzás
A=C , B=T
Osztás előkészitése
C=A/T
Osztás
I$ = 'I'
C kiiratása
n
P=C
Mentés
A=T , B=1
Összeadás előkészitése
C=A+B
T növelése
T=C
A=V , B=C
Kivonás előkészitése
C=A-B
V csökkentése
V=C

Számok törzstényezős felbontása
/PRIMPROG/

Az itt leirt program maximálisan 119 jegyü pozitiv egész számok törzstényezős felbontását végzi el. Meghatározza a szóban forgó szám törzstényezőit nagyság szerinti sorrendben, s azokat multiplicitásukkal együtt kiirja a képernyőre.

A jegyek számára tett korlátozás abból adódik, hogy itt a számot egy INPUT utasitással töltjük be az A$ karakterláncba, s az ABC-80 tipusu számitógép egy INPUT utasitás hatására maximálisan 119 karaktert képes elfogadni. Valamivel bonyolultabb beolvasási módot alkalmazva ez a korlátozás gyakorlatilag megszüntethető, természetesen ekkor is jelent a memóriakapacitás és a futási idő.

A program müködése, amit a mellékelt blokkdiagram is szemléltet, a következő. Az A$ decimális karakterláncba beolvasott A nemnegativ egész számot a KARTOM nevü rutin aritmetikai tömbbé alakitja, és az A%(*) tömbbe helyezi át. Ha a szám 0 vagy 1, akkor törzstényezős felbontása nincs, a program kiirja magát a számot a képernyőre, majd ujabb felbontandó szám begépelésére vár.

1-nél nagyobb számok esetén két, egymásba ágyazott ciklus segitségével keressük a szám törzstényezőit, s azok multiplicitását.

A külső ciklus egyes lépéseiben rendre a 2, majd az azt követő páratlan számok sorozatát vizsgáljuk abból a szempontból, hogy osztói-e az

$$X = \frac{A}{p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_i^{\alpha_i}}$$

számnak, ahol $p_1, p_2, \ldots, p_i$ az A szám korábbi ciklusokban talált primtényezői, $\alpha_1, \alpha_2, \ldots, \alpha_i$ pedig azok multiplicitásai. /Az első ciklus elején X = A./

A belső ciklus a talált törzstényezők multiplicitását határozza meg.

Tegyük fel, hogy a program a külső ciklus korábbi lépései során a fenti sorozatban szereplő és a B-nél kisebb számokat már megvizsgálta, s most, a következő ciklislépésben éppen a B-t vizsgálja. Az előbbi feltevés miatt az X számnak nincs B-nél kisebb törzsosztója.

A T számlálót /a multiplicitás számlálója/ a külső ciklus elején 0-ra állitjuk, majd az X és B számon a DIVRUT nevü rutin segitségével maradékos osztást végzünk. Ha a maradék 0, akkor B osztója X-nek /és igy A-nak is/ és B nyilván törzsszám, /$p_{i+1} = B$/ különben ugyanis X-nek /és igy A-nak is/ lenne B-nél kisebb, de $p_1, p_2, \ldots, p_i$-től különböző törzsosztója, ez pedig ellentmondás. Ebben az esetben T értékét megnöveljük 1-gyel, majd a kapott hányadoson - amit ujra X-szel jelölünk - és ugyancsak a B számon ciklusban /belső ciklus/ mindaddig végezzük a maradékos osztást, és a T 1-gyel való növelését, amig csak a maradék 0-tól különböző nem lesz. /Ez természetesen már az első lépésben is előfordulhat./ T > 0 esetén B törzsszám, X legkisebb 1-nél nagyobb osztója, s T értéke ekkor azt mutatja, hogy ez a B törzsszám mekkora multiplicitással szerepel a

$$\frac{A}{p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_i^{\alpha_i}}$$

számnak, és igy egyben A-nak a törzstényezős felbontásában. Ekkor a program a B törzsszámot és annak T multiplicitását kiirja a képernyőre. T = 0 esetben B nem osztója X-nek és igy A-nak sem. Ha az utolsó 0 maradékot adó osztásnál fellépő hányados 1, akkor a szám törzstényezős felbontása befejeződik, s a gép uj felbontandó szám begépelésére vár. Ellenkező esetben azt vizsgáljuk meg, hogy az első pozitiv maradékot adó osztás során fellépő hányados kisebb-e, mint a B osztó. Ez esetben az X osztandó primszám, ugyanis ha lenne B-nél nagyobb osztója, akkor annak a társosztóját már előbb megkaptuk volna X osztójaként. Ekkor a program kiirja az X osztandót, mint törzsosztót és az 1-et, mint annak a

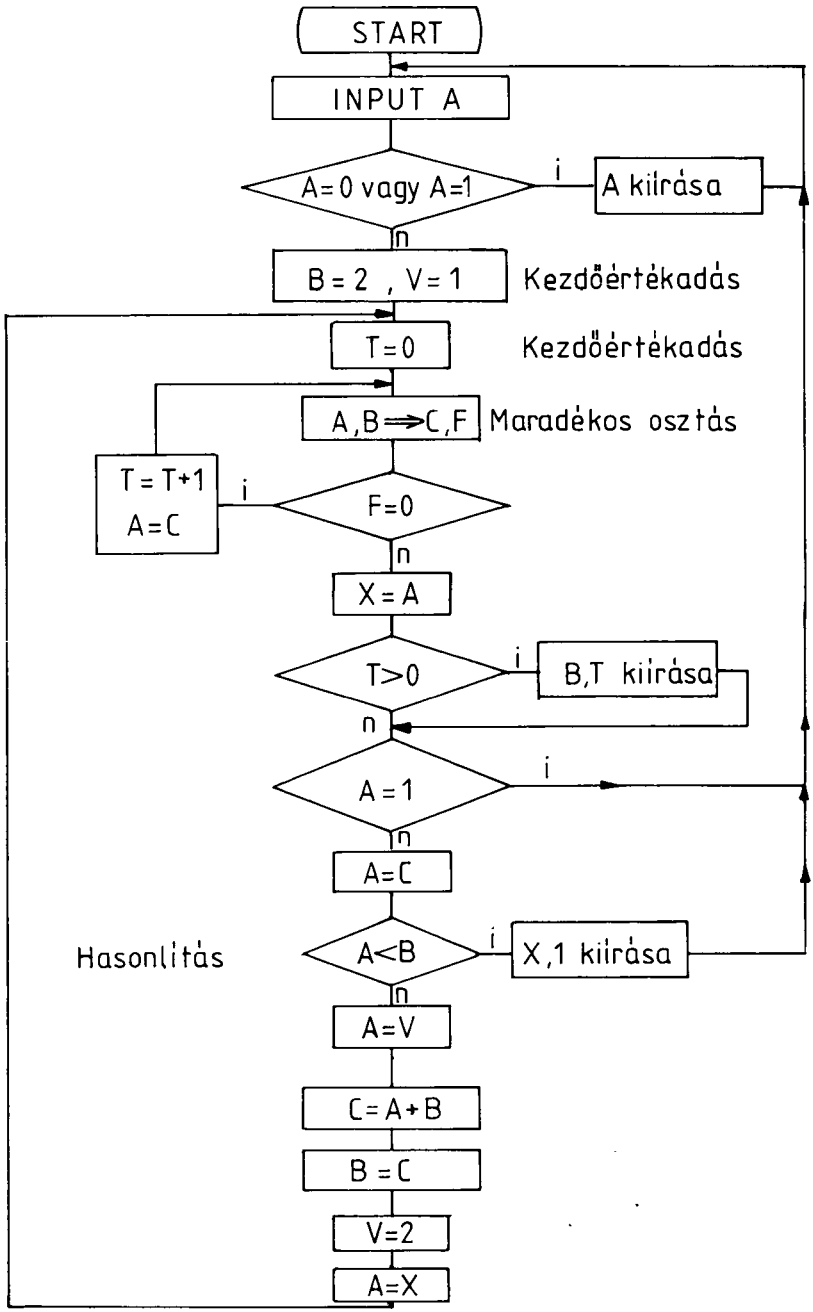
multiplicitását, majd uj szám begépelését várja.

Ha viszont ez sem következik be, akkor a B = B+V, a V = 2 és az A = X értékadás után a program a ciklus következő lépésével folytatódik.

A blokkdiagram jól mutatja, hogy a program elején végzett B = 2 és V = 1, valamint a ciklus végén lévő B = B+V és V = 2 értékadás biztositja, hogy a program rendre a fentemlitett sorozat tagjait vizsgálja a ciklus egyes lépéseiben. Megemlitjük, hogy a 2-nél nagyobb páros számokat a futási idő csökkentése céljából hagytuk ki a vizsgálatból, mint feleslegest. A ciklus bonyolultabb szervezése árán a futási idő még tovább csökkenthető.

A program hivja az ADDRUT, DIVRUT és COMPRUT nevü rutinokat, ezeket tehát futtatáskor be kell tölteni.

A program egy-egy szám felbontása után mindig ujabb szám betöltését várja, futását a CTRL és C billentyük együttes lenyomásával szakithatjuk meg.


START
INPUT A
A=0 vagy A=1
i
A kiírása
n
B=2 , V=1
Kezdőértékadás
T=0
Kezdőértékadás
A,B⇒C,F
Maradékos osztás
T=T+1
A=C
i
F=0
n
X=A
T>0
i
B,T kiírása
n
A=1
i
n
A=C
Hasonlítás
A<B
i
X,1 kiírása
n
A=V
C=A+B
B=C
V=2
A=X

SMALL COMPUTERS IN SECONDARY SCHOOLS

/Construction and some applications of an arithmetic/

by

Dr. József Csuri

Summary

The first part of the article analyses the tasks and possibilities of secondary schools from the aspect of the widespread and rapid growth of use of computers. It touches on the role of computers as ancillary aids in education, and on the resulting tasks falling on secondary school teachers.

The second part of the article describes subroutines in BASIC language for an ABC-80 computer, eliminating the limits resulting from the mode of number depiction of this machine; with these subroutines, addition, subtraction, multiplication, residual division and raising to a power can be performed with complete accuracy even for "very-high" non-negative integers. The restriction to non-negative integers does not mean any essential limitation.

With the aid of the subroutines, further subroutines are described that are suitable for calculation of quantities frequently occurring in number theory and combinatories, and an independent program is presented as an illustration of the construction of the program from subroutines.